DOCUMENT RESUME

ED 087 377 IR 000 046

AUTHOR McGinnis, N. P.
TITLE Macro-Graphics for the 1500 System.
INSTITUTION Alberta Univ., Edmonton. Div. of Educational Research.
PUB DATE Aug 72
NOTE 20p.; Paper presented at the ADIS Conference (Cap-Rouge, Quebec, Canada, August 8 through 10, 1972)

EDRS PRICE MF-$0.65 HC-$3.29
DESCRIPTORS Comparative Analysis; *Computer Assisted Instruction; *Computer Graphics; Computer Programs; Computer Science; *Display Systems; Higher Education; Instructional Systems; Programing; Systems Development; Technological Advancement; Time Sharing
IDENTIFIERS Cathode Ray Tube; Coursewriter; CRT; Dictionary Graphic Capability; DRAW; IBM 1500; *Macro Graphic System; Universal Dictionary

ABSTRACT

The 1500 System allows only a restricted number of dictionary-graphic areas in core. To solve the practical problems thereof, the Division of Educational Research Services at the University of Alberta attempted to use a universal dot dictionary, but this solution was not feasible for two reasons: Coursewriter input cannot exceed 250 characters per statement and a dot dictionary is a combersome solution for authors. An emphasis therefore was put on developing the Macro-Graphic System. This system has two most important features. They are: 1) any graphic display can be set up on the CRT without the need for dictionary core-areas as set up in the 1500 System; 2) it is easier for an author to learn to use this system than it is for him to master the existing 1500 dictionary graphic system. In addition, more courses can be supported simultaneously on this system. A Coursewriter course, called DRAW, along with two special purpose functions DRAWX and DRAWY have been developed to prepare macro-graphics. (CH)

MACRO-GRAPHICS FOR THE 1500 SYSTEM

N.P. McGinnis
Division of Educational Research Services
The University of Alberta
Edmonton, Alberta

ADIS Conference
August 8-10, 1972
Quebec City. P.Q.

ABSTRACT

The purpose of this paper is to introduce you to the work being done in the Division of Educational Research Services at The University of Alberta. Emphasis is placed upon graphic displays for the IBM 1500 CAI System.

I would first like to outline our System to give you an insight as to why we developed the Macro-Graphic System. Ours is an 1130 based System with five disks and two tape drives (see Figure 1 in Appendix). There are sixteen student stations, each consisting of a crt, keyboard, light pen, audio unit and film projector (see Figure 2 in Appendix). There are two additional typewriter terminals.

To date, most of our research projects have been at the initiative of graduate students wanting to carry out studies specified in their thesis topics. These projects have therefore been of short duration. As a result most authors did not develop any authoring expertise until their projects were near completion. It has been the job of the operating and systems staff to ease these students through and hopefully avoid major complications.

As students developed their programs they of course attempted to utilize all of the bells and whistles offered by the 1500. One of the bells rung the hardest was the dictionary-graphic capability. Naturally everyone deviated from the systems dictionary and eventually had to make use of graphics. I am not being critical at this point since presumably good courses should make use of visual techniques. I am however speaking as a systems person. Most of you who are familiar with the 1500 will realize that as time passed we had acquired courses which were mutually exclusive to each other. This resulted because the 1500 System was designed to allow a restricted number of dictionary-graphic areas in core (a maximum of twelve). Once this limit is exceeded, people wanting to sign-on to courses with specified dictionary-graphics have to wait until others utilizing the dictionary areas, do in fact, sign off.

Hence some courses become mutually exclusive to each other. Systems people could usually predict the storm to come and would send out warnings. Unfortunately these warnings were usually not heeded by the uninitiated author. Schedules, etc., had to be set up to accommodate everyone. To be more positive, however, most projects did run up to completion, graduate students received their degrees and we prepared ourselves for new authors, determined to avoid problems experienced in the past.

Coupled with the problem of a restricted dictionary-graphics area, is the fact that in the last four years we have probably averaged two demos per week (see Figure 3 in Appendix). This is to say that about seven to ten thousand persons have had some experience with our System. One portion of the demo program is a small collection of research projects developed by graduate students. The other portion of the demo consists of programs received from other 1500 Installations as well as programs especially developed for demonstration purposes. It took considerable time and effort on the part of our System Coursewriter programmer to meld these programs together into a coherent package. It took even more effort to reduce the number of dictionary-graphic areas to nine. But we still had the problem that if the demo was on, other programs could not be run concurrently. This was rather difficult to explain to visitors especially after informing them that this was a timesharing system capable of allowing each terminal to execute a different course.

These were a few of the points that made us look for a viable solution. The first solution attempted was a universal dictionary, which would contain the most frequent 8 by 12 dot sub-sets. It did

not take too long to discover that the optimal solution was the "dot dictionary" with a few extra multi-dot characters thrown in. We felt this solution was not feasible for two reasons:

1) Coursewriter input cannot exceed 250 characters per statement. This means that some "display text" statements cannot be properly completed.

2) Utilizing a dot dictionary is a very cumbersome solution for authors. Remember that we assume very little expertise on the author's part, and we are in fact trying to encourage not discourage CAI.

At that point in time we admitted temporary defeat. Authors continued to implement more and more dictionary graphics sets. Authors were warned of the implications, but even we proved to be abetting the crime. We had developed a program for documenting dictionary and graphic sets. The program had, as printer output, the dot representations of characters and graphics (see Figures 4 and 5).

Figure 4.
Printer Output of Dictionary Characters

Figure 5
Printer Output of Graphic Drawings

This program helped avoid duplication of characters and graphics as new authors could now check to see what had been developed before. We even went a step further. A program was created that allowed a user to create dictionary characters on-line. The main intent of this program was to avoid the keypunching frustrations associated with the production of dictionary characters.

Today, however, we have a solution to our problem. It is the Macro-Graphic System . The two most important features of this system are:

1) Any graphic display can be set up on the crt without the need for dictionaries or graphic sets. In short the system avoids the use of dictionary core-areas as set up in the 1500 System.

2) It is easy (or easier) for an author to learn to use this system than it is to master the existing 1500 dictionary graphic system.

Let me first explain how the system eliminates the need for dictionary areas in core. Assume that an author is coding a course on-line and that he has already created the macro-graphics that he requires. In fact the macto-graphics are stored in the on-line macro-file. To insert the macro-graphic, the author must issue a regular "call macro" statement (see Figure 6).

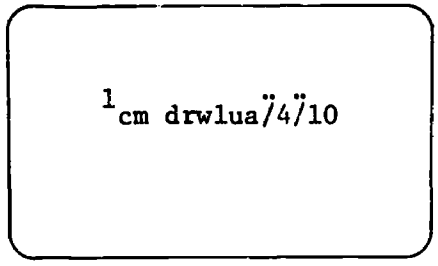


Figure 6.

Use of Call Macro by Author to Obtain Graphic.

The macro name will have "drw" as the first three characters followed by three characters defined by the system when the macro was created. Only two additional parameters are required. These are the top-most row and the left-most column where the graphic is to be displayed. The row limits are 0 to 30 and the column limits are 0 to 38. Both parameters may be specified as integers, counters, or defaulted. The action will be the same as a normal "dt" statement, i.e., the display will start on the next row in column 0. Given that the author made no mistakes in calling the macro he

is able to immediately execute the sequence and have the graphic appear on the crt. If the author was to type out the macro call, he would encounter Coursewriter coding which would appear as in Figure 7.

```
ty draw-2⁄draw-29

 2  no  drwlua⁄4⁄10*e

 3  fn  dq⁄4,10,00*e

 4  no  *e

 5  fn  dq⁄,10,00*e
 6  no  *e
 7  fn  dq⁄,10,00*e
 8  no  *e
 9  fn  dq⁄,10,00*e
10  no  *e
11  fn  dq⁄,10,00*e
12  no  *e
13  fn  dq⁄,10,00*e
14  no  *e
15  fn  dq⁄,10,00*e
16  no  *e
17  fn  dq⁄,10,00*e
18  no  *e
19  fn  dq⁄,10,00*e
20  no  *e
21  fn  dq⁄,10,00*e
22  no  *e
23  fn  dq⁄,10,00*e
24  no  *e
25  fn  dq⁄,10,00*e
26  no  *e
27  fn  dq⁄,10,00*e
28  no  *e
29  no  em*e
type control word
```

Figure 7.

Coursewriter Macro Code for Graphic.

You will note the macro consists of statements calling a function named "dg," followed by no-op statements.

The dg function operates using the following logic sequence. The row and column coordinates are tested for acceptable limits. If either coordinate is defaulted, a default coordinate is inserted. If either parameter is a counter, the content of that counter is loaded and testing continues. The next parameter is defined at macro definition time. This number, times two, will give the column width of the graphic to be displayed. The number also indicates whether one or two no-op statements follow: Should the appropriate number of no-op statements not follow in sequence then the dg function returns the error code "dg" to the interpreter.

Given that the no-op statement follows in sequence, it is the job of the dg function to load the data supplied in the no-op statement into the I/O buffer pool. The loading of data from the dg function and no-op statements to the I/O buffer pool occurs in the same sequence as it does when the interpreter executes action for "dt" or "dg" commands. Once the data are in the buffer pool, interrupt level I/O routines will execute and send the data to the appropriate crt terminal. In the normal sequence this involves both translation and transfer of data (see Figure 8).

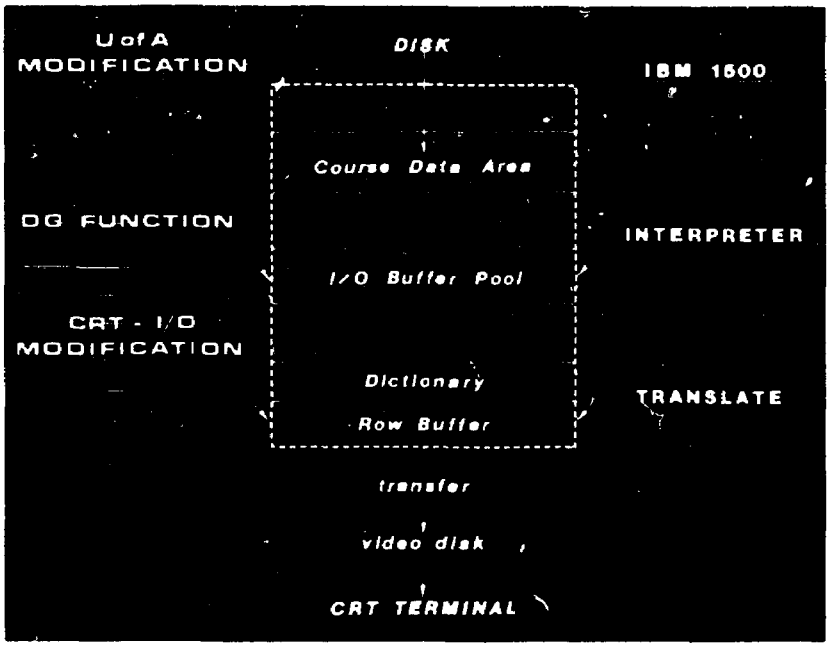


Figure 8.
Execution Steps for Macro Graphic System.

Translation, in the 1500 System, means that the real data to be displayed on the crt must first be retrieved from the appropriate in-core dictionary.

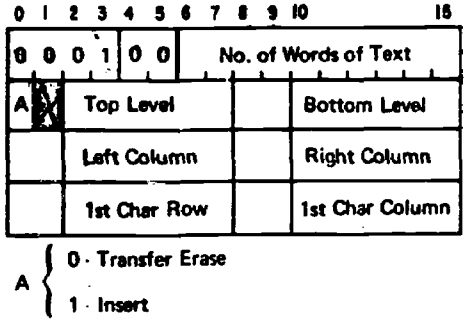
Data that form the "dt" statement are in reality a set of pointers to the data in the dictionary. The translation feature uses the pointer data to retrieve and store dictionary data in the "row buffer." Then, a transfer instruction moves the data from row buffer to the specified video disk track in the 1502. With the next revolution of the video disk the data will appear on the specified crt.

In our system one major difference exists. As the data is being loaded to the I/O buffer pool, the second word of the crt I/O command will have bit position 1 turned on (see Figure 9).

DEVICE-ORIENTED STATION I/O COMMANDS

1510 Instructional Display

OUTPUT Station I/O Command



This command causes lines of characters to be displayed on the 1510 Instructional Display. The command requires four words plus n words of associated text that contains 2n or 2n-1 EBCDIC characters. The last character must be an EOB. The first six bits of the command contain the command identifier. The remainder of the command contains eight control parameters as follows:

- Number (n) of words to be displayed. This parameter (word 1, bits 6-15) may have the range 0 to 1023.

Figure 9.

Station I/O Command Format.

The station I/O control routines have been modified to test the status of this bit. If the bit status is off, the previously described translation-transfer process will occur. If the bit is on, then the translation process will be substituted by another routine. This routine moves the data from the I/O buffer pool to the row buffer. Once this task is complete the routine then returns to the normal processing logic where the transfer of data from the row buffer to the video disk occurs (see Figure 8). This simple procedure avoids the use of dictionary areas.

It should be admitted at this point that the system can put a load on the buffer pool. We are no longer making use of the pointer system which expands data from a one byte pointer to six words of real data (a dictionary character is six words long). Our system starts with the original six words. The six words are carried from the course (on disk) into the course-data area in core, then to the I/O buffer pool and finally to the row buffer. Therefore, we have twelve times as much data.

Although the modification of the translation feature is ten times slower in the execution phase and more data must be moved to various core locations we have been able to reduce to two the number of dictionary areas required for our demo. This has significantly increased the number of available function areas. The total result, using gross measurements, has been a reduction in the system latency time from about six seconds down to about two seconds. For our purposes system latency is defined as the amount of time elapsed between a student response and any form of feedback to the student from the system.

Since there is less referencing to disk for required functions our initial reaction has been that the disk system is a potential bottleneck. We have conducted tests which reaffirmed this belief.

Another obvious advantage of the macro-graphics approach is that more courses can now be supported simultaneously on our System. For example, we have a course designed to teach deaf children to read and write. Extensive use is made of graphics. This course could not have been done with dictionaries since at least twenty dictionaries would have been required at all times.

I would now like to turn your attention to the problem of preparing macro-graphics. A Coursewriter course, called DRAW, along with two

special purpose functions DRAWX and DRAWY have been developed to prepare macro-graphics. Both functions incorporate the facility of loading the special dt command to the I/O buffer pool. The DRAW course makes use of a file with an allocation of sixteen sectors of disk space for each crt terminal. It is in this disk space that the graphic being developed is saved until completion.

When a user signs-on to DRAW he will be presented with the graphic list constructed at that terminal (see Figure 10 in Appendix). Because the user is able to begin modifying the graphic as presented we have no security precautions against destroying unfinished work. Initially the user has the option of:

1) erasing part or all of the graphic.

2) copying part of the graphic from one part of his screen to another. (He also has the option to copy material produced at another terminal.)

3) copying inverse portions of the graphic on the screen: [he can copy material and have it rotated 180 degrees on a horizontal or vertical axis (see Figures 11 and 12)].

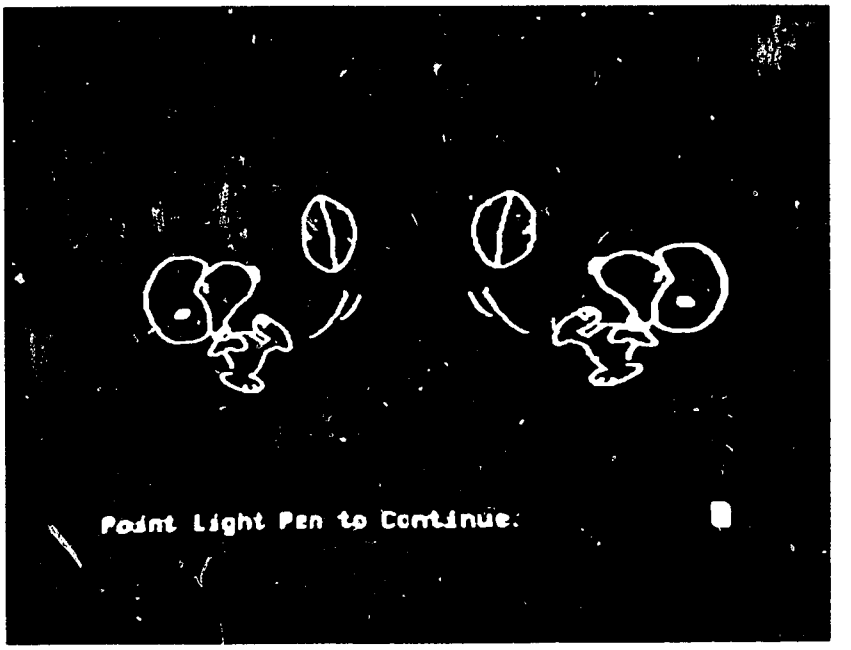


Figure 11.
180° Rotation on X axis.

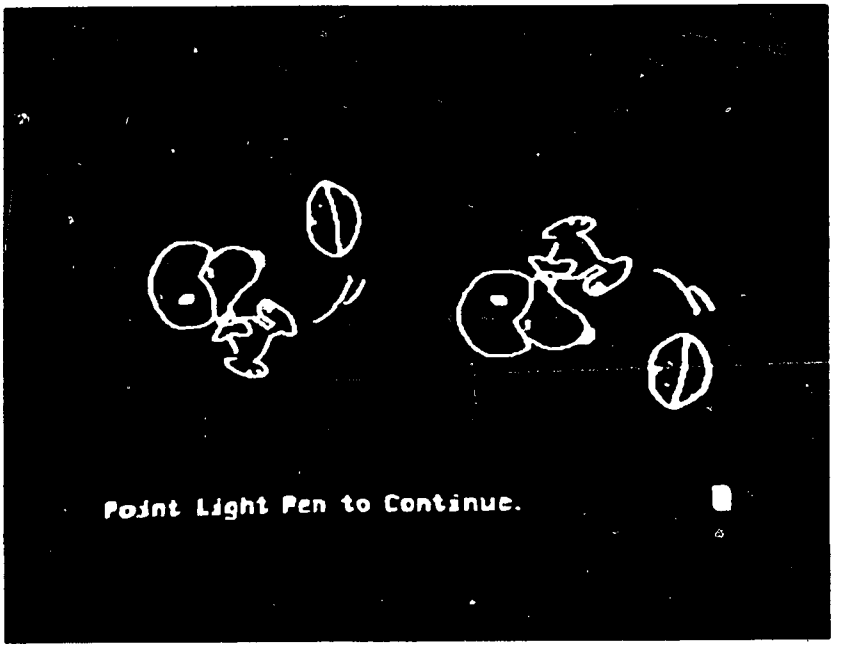


Figure 12.
180° Rotation on Y axis.

4) viewing the graphic as the end product.

5) saving the graphic by specifying the boundaries of his graphic (see Figure 13).

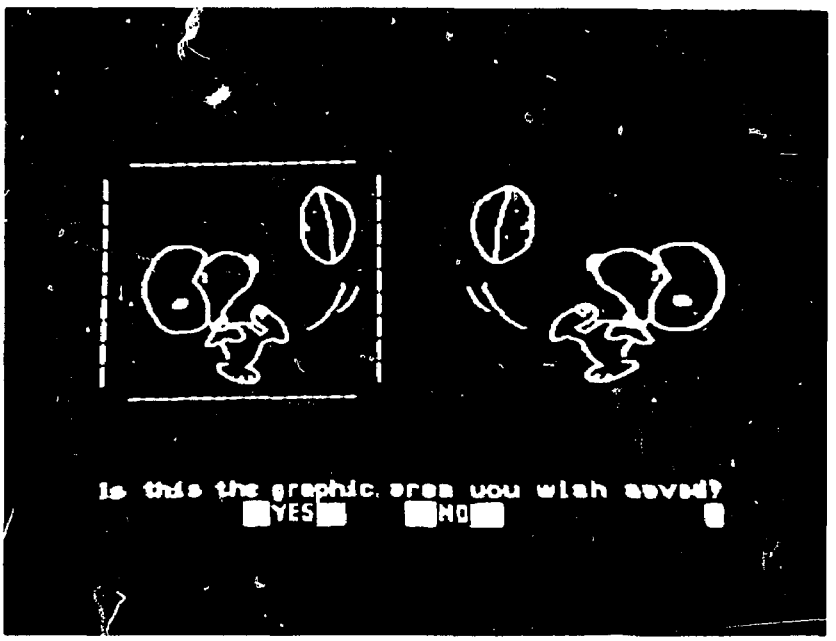


Figure 13.

Saving Graphic Within Defined Bounds.

The lower limit is a graphic two rows high by two columns wide. The upper limit is a graphic twenty-eight rows high by forty columns wide. The system will specify a three character identifier to be concatenated to the DRW characters to form the name of the macro-graphic.

During course execution performance recordings are taken. During background time utility routines retrieve the graphic information from the performance recording file and store it in the macro file.

6) pointing anywhere on rows 0 to 27. The character position pointed to will be blown up. That is, the author will be presented with an expanded copy of the 8 by 12 dot matrix pointed to (see Figure 14).

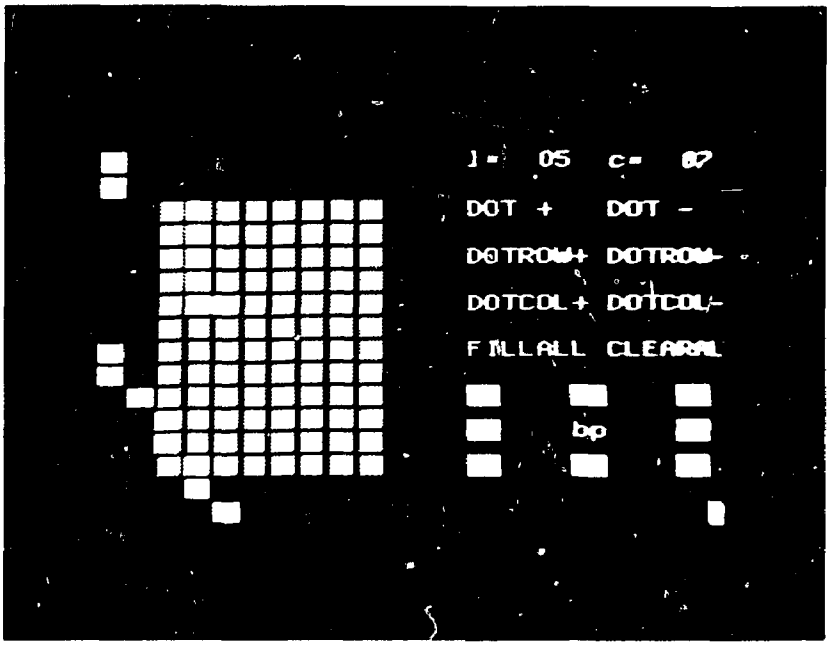


Figure 14.
Expanded Dot-Matrix of Selected Graphic Area.

In addition the dot row adjacent to the top and bottom of the character will be blown up as well as adjacent bit columns to the left and right of the character. This provides the user with some continuity to visualize the graphic being developed. By utilizing the light pen, the user is then able to re-define the character to his specifications. Once this is done the character can be returned to the original graphic in its new representation and the user has the option of either displaying the total graphic or displaying a blow-up of any adjacent 8 by 12 dot matrix.

The authors have been very responsive to the macro-graphic system. Some minor suggestions have been incorporated, and at present we are looking at a system for documenting the macro-graphics already produced. One alternative is to utilize the plotter on The University of Alberta's IBM 360/67, to produce the image.

During the past year a number of junior and senior high school students have learned Coursewriter II. One of the high school students also learned 1130 Assembler and wrote a very useful function for the Macro-Graphic System. We refer to the product as the "expand" (or magnification) function. The function makes use of the crt I/O modification routines. The function has parameters analogous to the "dti" statement where the starting row and column are specified along with data to be presented. If a buffer is specified, the function displays the contents of the buffer. One additional parameter is required. This alerts the function to the magnification factor. For example, if the magnification factor is four, then data specified which would normally appear on two rows, will now be expanded to occupy eight rows, and four times as many columns. Magnification is allowable up to and including 99 times. However, when the expansion size is five or more the output occasionally appears somewhat distorted (see Figure 15).

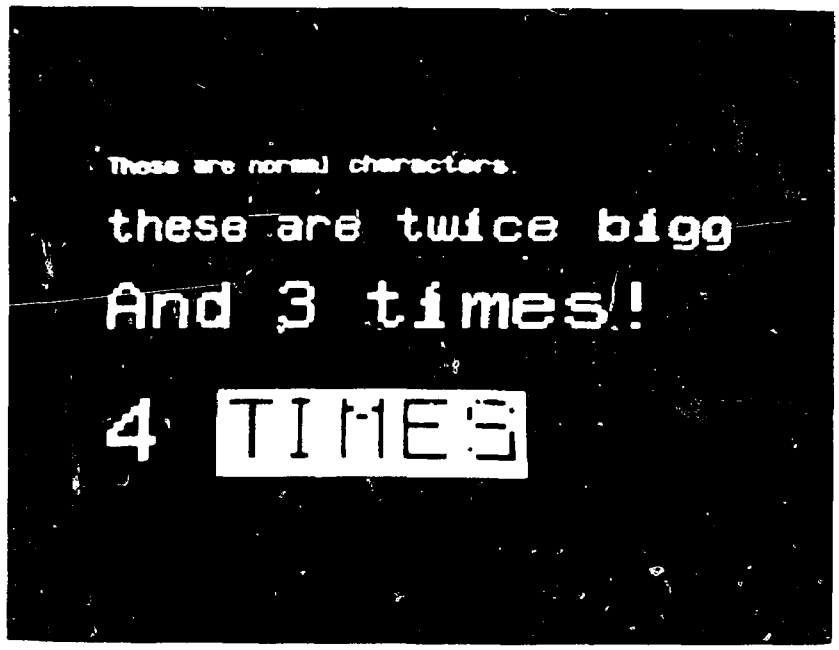


Figure 15.
Example of "Expand" Function.

The magnification factor has proved very useful for feedback messages or emphasis.

Both the macro-graphic and magnification of text facilities were designed to provide adequate graphics for all levels of instruction. Our modified system makes the IBM 1510 crt a unique and effective electronic blackboard as well as recuding system response time. Now an author can simply treat graphic and non-standard graphic representations as textual material. For example, with the magnification facility, a CAI author can easily vary the text size for young children.

We have also explored the possibility of using Coursewriter III. However, because of the inadequate terminal devices supported by

Coursewriter III, when compared to those on the 1500 System, we doubt that our users would shift from the 1500 System.

APPENDIX

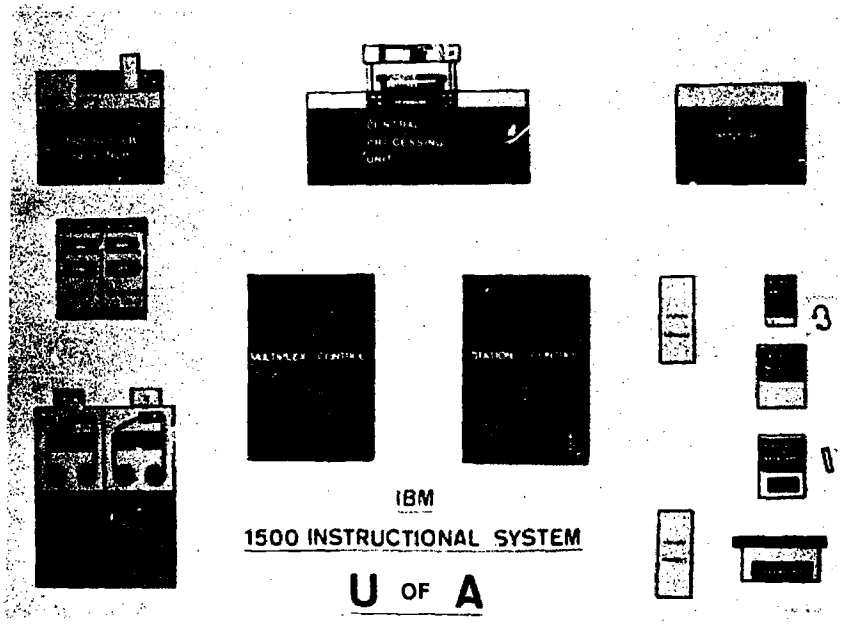


Figure 1.
The University of Alberta IBM 1500 Configuration

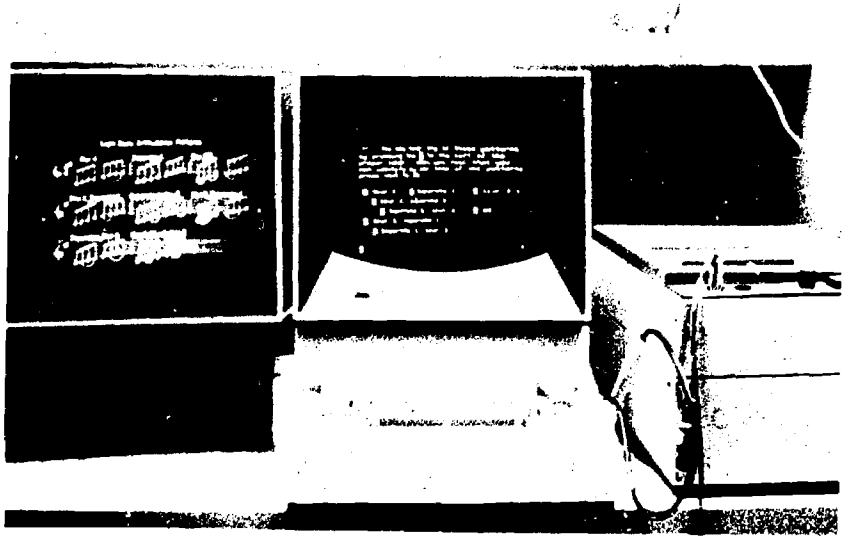


Figure 2.
IBM 1500 Terminal Configuration

APPENDIX

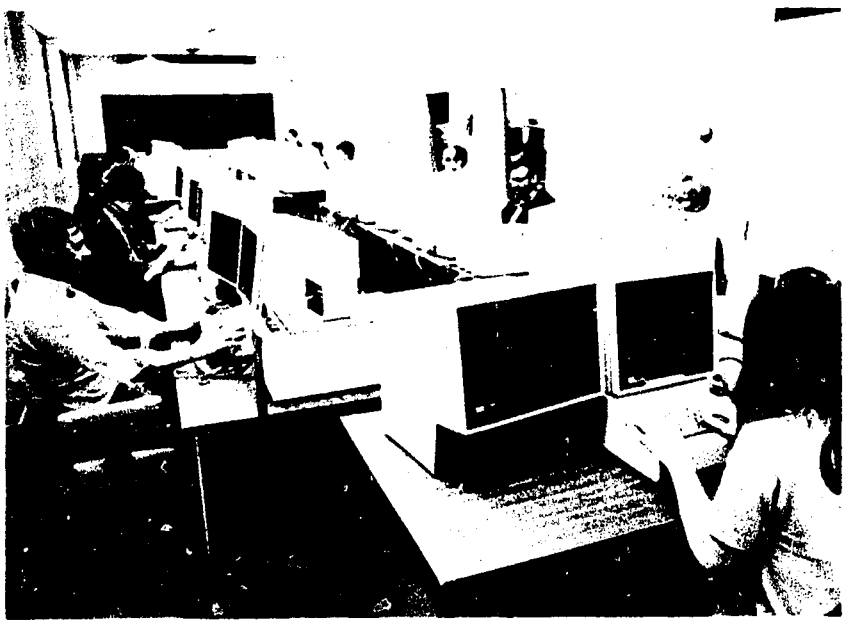


Figure 3.
The University of Alberta Terminal Room

Figure 10.
Construction of Graphic Using Light Pen